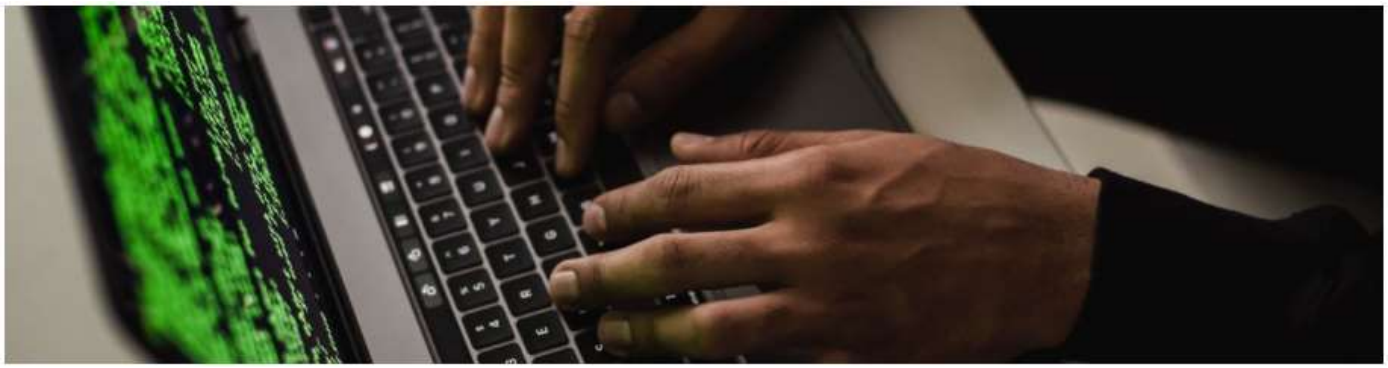
¿Qué es encapsulamiento?

Alex Felipe
04/11/2021



En mi sistema de inscripción de libros yo necesito del nombre y del precio:

```
public class Libro {

    private final String nombre;
    private final double precio;

    public Libro(String nombre, double precio) {
        this.nombre = nombre;
        this.precio = precio;
    }
```

```
    //métodos
}
```

Para inscribir un libro yo necesito pasar las informaciones y enviar para el banco de datos:

```
Libro libro = new Libro("Java", 27.83);

Conexion conexion = hagaConexionConElBancoDeDatos();
conexion.registraLibro(libro);
conexion.cierraConexion();
```

Todas las veces que yo quiera inscribir un libro yo tendré que instanciarlo, tomar una conexión, guardar el libro y cerrar la conexión. Vamos a suponer que ahora nosotros necesitamos verificar si el libro ya existe en el banco, y en caso de que exista, necesitamos actualizar en vez de guardar:

```
Libro libro = new Libro("Java", 27.83);

Conexion conexion = hagaConexionConElBancoDeDatos();

if(!conexion.existeLibro(libro)){
    conexion.registraLibro(libro);
} else{
     conexion.actualizaLibro(libro);
}
conexion.cierraConexion();
```

Cuantos más procedimientos, necesitamos para poder guardar el libro, mayor quedará la rutina de guardar libros. Imaginate todas las clases que necesitan guardar un libro, tendrán que ser actualizadas todas las veces que hubiera un cambio! Es una pésima solución... ¿Que tal cambiar esos procedimientos de guardar el libro para un único método? Entonces, quedaría:

```
Libro libro = new Libro("Java", 27.83);
```

```
guardaLibro(libro);

public void guardaLibro(Libro libro){
     Conexion conexion = hagaConexionConElBancoDeDatos();
     if(!conexion.existeLibro(libro)){
         conexion.registraLibro(libro);
    } else{
        conexion.actualizaLibro(libro);
    }
    conexion.cierraConexion();
}
```

Ahora no necesitamos saber sobre conexión o cualquier procedimiento para guardar un libro, apenas llamamos el método `guardaLibro()`, enviamos el libro, y él hace todo por nosotros! cuando transformamos un procedimiento que presenta muchas reglas de negocio o rutinas complejas en un método, llamamos eso de **encapsulamiento**.¿Y si tuvieramos que hacer un *backup* de los libros cada vez que guardamos o actualizamos? Bastaría agregar ese trecho en un único lugar, o sea, en el método `guardaLibro()`:

```
//continuo usando el guardaLibro() de la misma forma que guardaLibro(libro);

public void guardaLibro(Libro libro){

    //rotina para guardar libro
    conexion.hacerBackupDelLibro();
    conexion.cierraConexion(); }
```

¡El código cambió y continuamos utilizando el método sin ninguna preocupación si él cambio o no! Además de eso, el código fue modificado en apenas un lugar, o sea, no necesitamos preocuparnos si va quebrar nuestro sistema en todos los otros lugares que usan ese método! Además de ser más elegante, nuestro sistema se convierte en mantenible.

Es siempre importante verificar en todos los puntos del código donde es posible aplicar el encapsulamiento, pues ganamos muchos beneficios:

- Mejor mantenimiento de código.
- División de responsabilidades.
- Reutilización de código.

¿Te gustó de encapsulamiento? ¿Estás listo para analizar tu código y verificar donde él puede ser aplicado? El encapsulamiento hace parte de una de las buenas prácticas de refatorización de código.

Preguntas Frecuentes:

¿ Qué es encapsulamiento?

Encapsulamiento es un principio de deseño de código, generalmente relacionado a programación orientada, que nos orienta a esconder las funcionalidades y funcionamiento de nuestro código dentro de pequeñas unidades (normalmente métodos y funciones). eso posibilita que modificaciones en el sistema puedan ser hechas de manera más quirúrgicas, sin que una funcionalidad esté dispersa por diversas partes del sistema.

¿Cuando usar encapsulamiento?

Básicamente siempre, pues la interface, la forma como clases y objetos conversan uno con el otro, debe siempre estar isolada de la forma como ejecutan lo que se propusieron hacer. Esa ejecución es la implementación del código. Como dice el libro Design patterns: programe volviendo a la interface, no a la implementación.

¿Qué es la quiebra de encapsulamiento?

Es cuando la implementación dio una funcionalidad 'fuga' para diferentes partes del sistema, con código en regiones, unidades, módulos o paquetes muy diferentes. De esa forma, siempre que necesitamos modificar esa funcionalidad, necesitamos alterar diversas partes distantes del código. Un ejemplo es el uso exagerado de variables globales: ellas comienzan a generar códigos dispersos que acceden y modifican esas variables, un fuerte acoplamiento, que amarra puntas que deberían estar más sueltas.

Alex Felipe Victor Vieira

Alex es instructor y desarrollador y posee experiencia en Java, Kotlin, Android. Creador de más de 40 cursos, como Kotlin, Flutter, Android, persistencia de datos, comunicación con Web API, personalización de pantallas, pruebas automatizadas, arquitectura de Apps y Firebase. Es experto en Programación Orientada a Objetos, visando siempre en compartir las buenas prácticas y tendencias del mercado de desarrollo de software. Trabajó 2 años como editor de contenido en blog de la Alura y hoy todavía escribe artículos técnicos.

Puedes leer también:

- Intercambiando caracteres de una String en Java
- ¿Cómo funciona el import y export de JavaScript?
- Diferencia entre int e Integer en Java
- Revisando la Orientación a Objetos: encapsulación de Java

ARTÍCULOS DE TECNOLOGÍA > PROGRAMACIÓN